16.412

Cognitive Robotics

Autonomous Visual Tracking Algorithms

Alexander Omelchenko

May 12, 2004

1. Problem Formulation

This paper describes an algorithm for autonomous visual tracking of a moving object. The algorithm is designed for a multi-vehicle surveillance system that includes a UAV and a ground rover. A typical mission scenario of such system is depicted in Figure 1. Visual coverage of the rover from the UAV is done in order to facilitate remote operation of the rover and to facilitate obstacle detection by the rover (as ground rovers, using their forward-looking cameras, are not very good at the detection of obstacles below

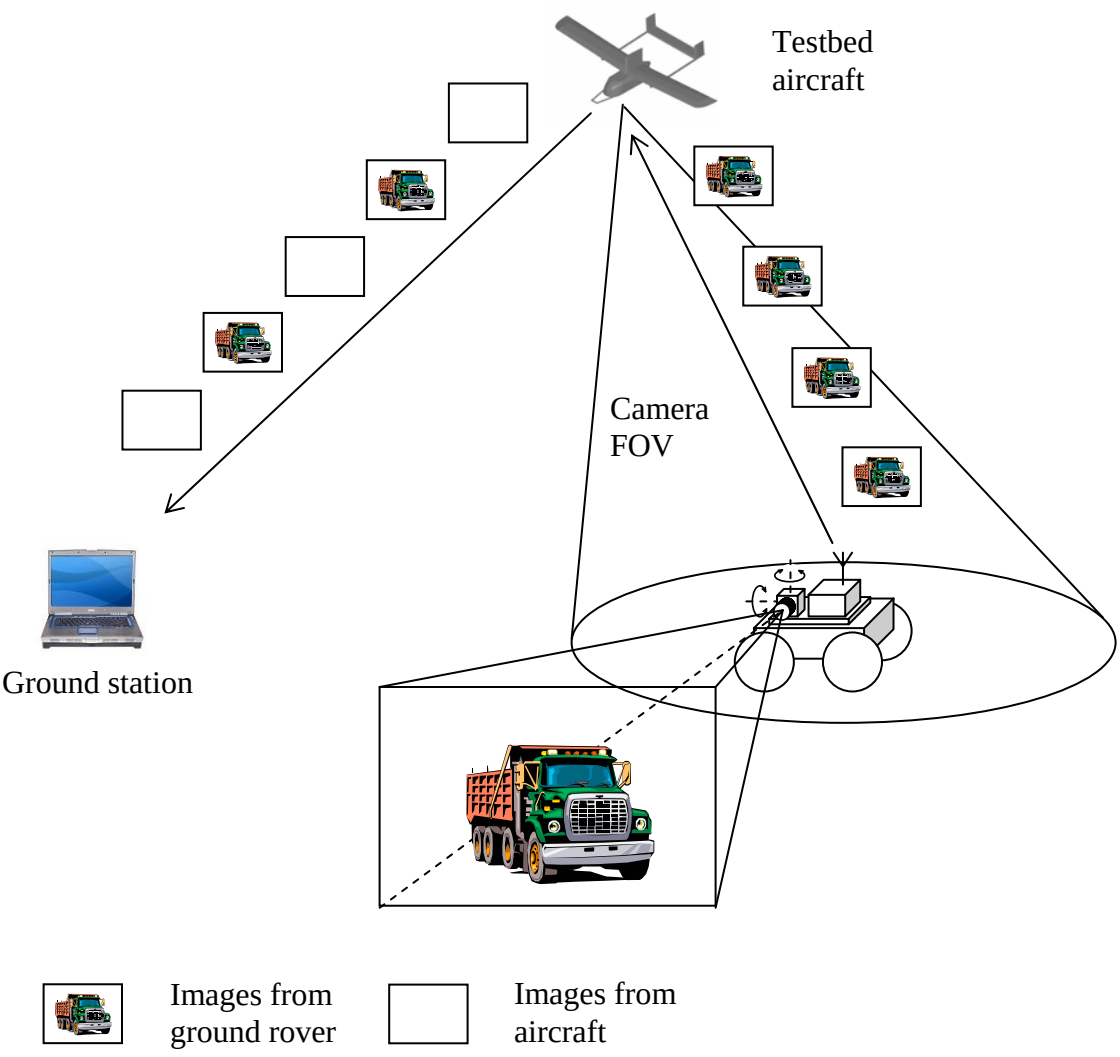


Figure 1: Mission scenario

ground level such as holes in the ground, and additional visual coverage from an aircraft allows to alleviate detection of such obstacles). At the same time, rover provides visual tracking of different objects on the ground. Both the aircraft and the rover are equipped with pan/tilt/zoom cameras.

General approach to the design of the control system is shown in Figure 2.

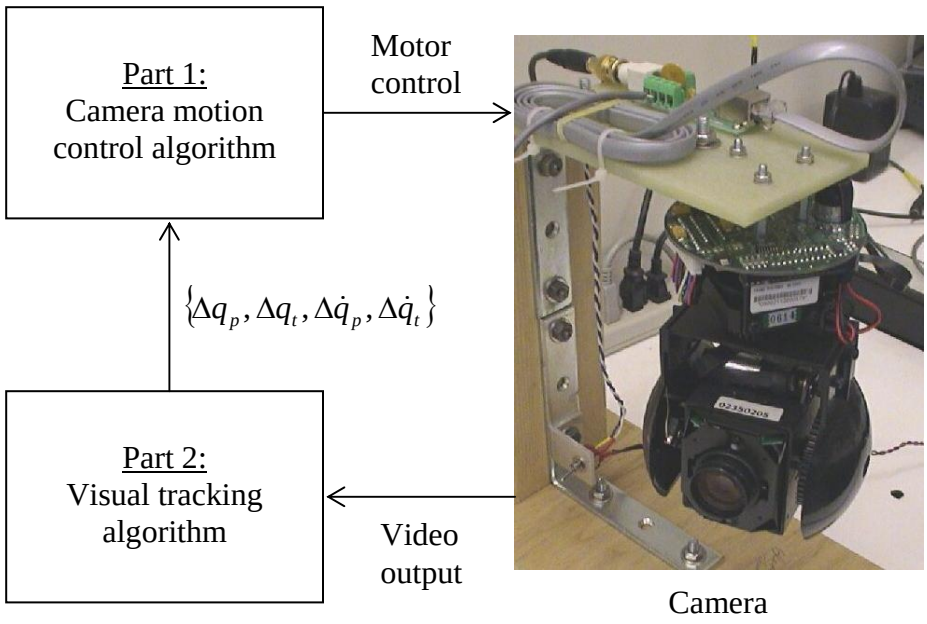


Figure 2: Control algorithm

The control algorithm will consist of two parts. Part 1 will generate control signals for the camera's motors, while part 2 will be doing visual tracking of a moving object in the field of view of the camera. The output of part 2 will consist of values of $\Delta q_p, \Delta q_t, \Delta \dot{q}_p, \Delta \dot{q}_t$, which are error signals of the pan and tilt angles of the camera and their rates of change, which will be used as an input to part 1 of the algorithm.

Before solving the objective problem depicted in Figure 1, a simplified version of the problem is considered. The experimental setup used in the simplified version of the problem is shown in Figure 3. Pan/tilt/zoom camera is mounted on the ceiling of a room, while the rover moves on the floor of the room. The problem for the camera is to do visual tracking of the rover.

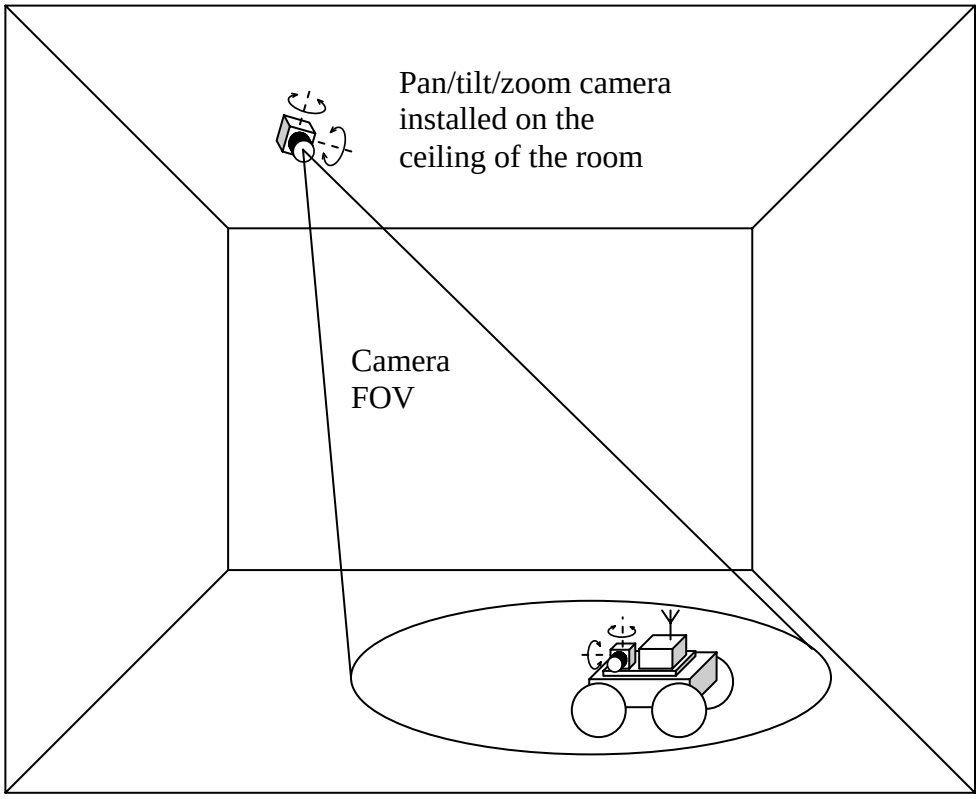


Figure 3: Experimental setup of simplified problem

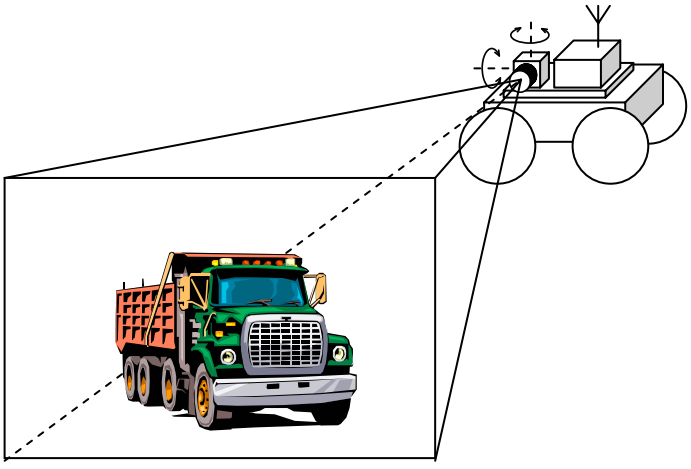


Figure 4: Tracking of moving objects from stationary rover

The experimental setup of Figure 3 is also equivalent to the situation in which visual tracking of a moving object is done from stationary rover, as depicted in Figure 4.

Part 1 of the control algrithm is outside of the scope of this class. Therefore this paper is focused on the second part of the control algorithm, i.e. the visual tracking part.

2. Different Types of Visual Tracking

As shown in Figure 5, there are two main types of visual tracking: continuous tracking and stepwise tracking. In the continuous tracking mode, camera always follows the object that is being tracked. One possible disadvantage of this mode is that the camera, making frequent adjustments to its orientation, may destabilize video.

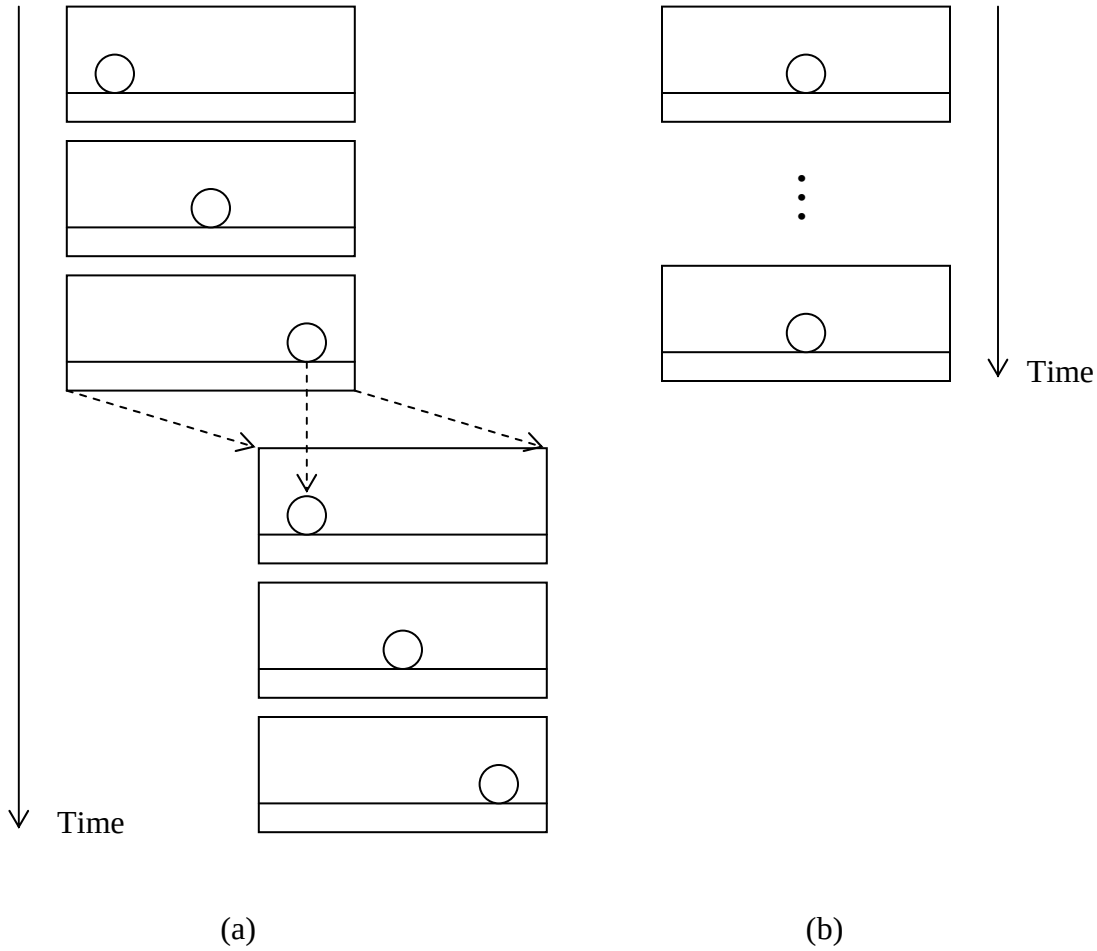


Figure 5: Visual tracking modes: (a) – stepwise tracking, (b) – continuous tracking

In the stepwise tracking mode, on the other hand, camera changes its orientation only at discrete moments of time with some period. During the periods of time between rotations, the camera's field of view stays fixed with respect to background.

3. Visual Tracking Algorithm One: Center of Mass Tracking

At the time of the writing of this paper, the experimental setup shown in Figure 3 is not ready yet. However, in order to test visual tracking algorithm, a sequence of images with any moving object is sufficient. Figure 6 shows a sequence of images that has been recorded from avionics computer of the testbed aircraft in the laboratory. The situation modeled in the images, with an object moving toward the camera is more challenging for visual tracking than a situation in which an object would move in a direction perpendicular to the line of sight. Details of the visual tracking algorithm are shown in Figure 7. Images (a) and (b) are two consecutive images. Image (c) is obtained by subtracting image (b) from image (a). Edge detection algorithm is applied to image (c). The result is binary image (d). Center of mass of image (d) is calculated using formulas (3.1) and (3.2).

$$M_{00} = \sum_x \sum_y I(x,y) \quad M_{10} = \sum_x \sum_y xI(x,y) \quad M_{01} = \sum_x \sum_y yI(x,y) \tag{3.1}$$

$$x_c = \frac{M_{10}}{M_{00}} \qquad y_c = \frac{M_{01}}{M_{00}} \tag{3.2}$$

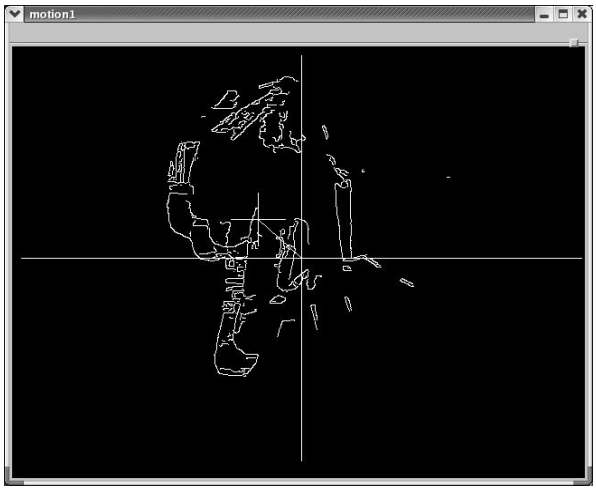
The result of the algorithm is image (e). Large cross indicates the location of the center of the image, while the small cross indicates the location of the center of mass of the object. Line connecting the two centers represents the error (because the purpose of the camera control algorithm is to keep the observed object in the center of the field of view). The results of the algorithm applied to the test sequence of images are shown in Figure 8 and Figure 9.

Figure 6: Test sequence of images

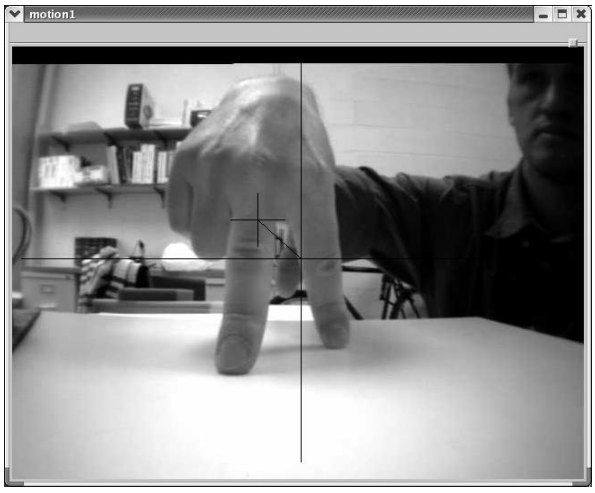
(a)

(b)

(c)



(d)



(e)

Figure 7: Visual tracking algorithm

Figure 8: Visual tracking algorithm results

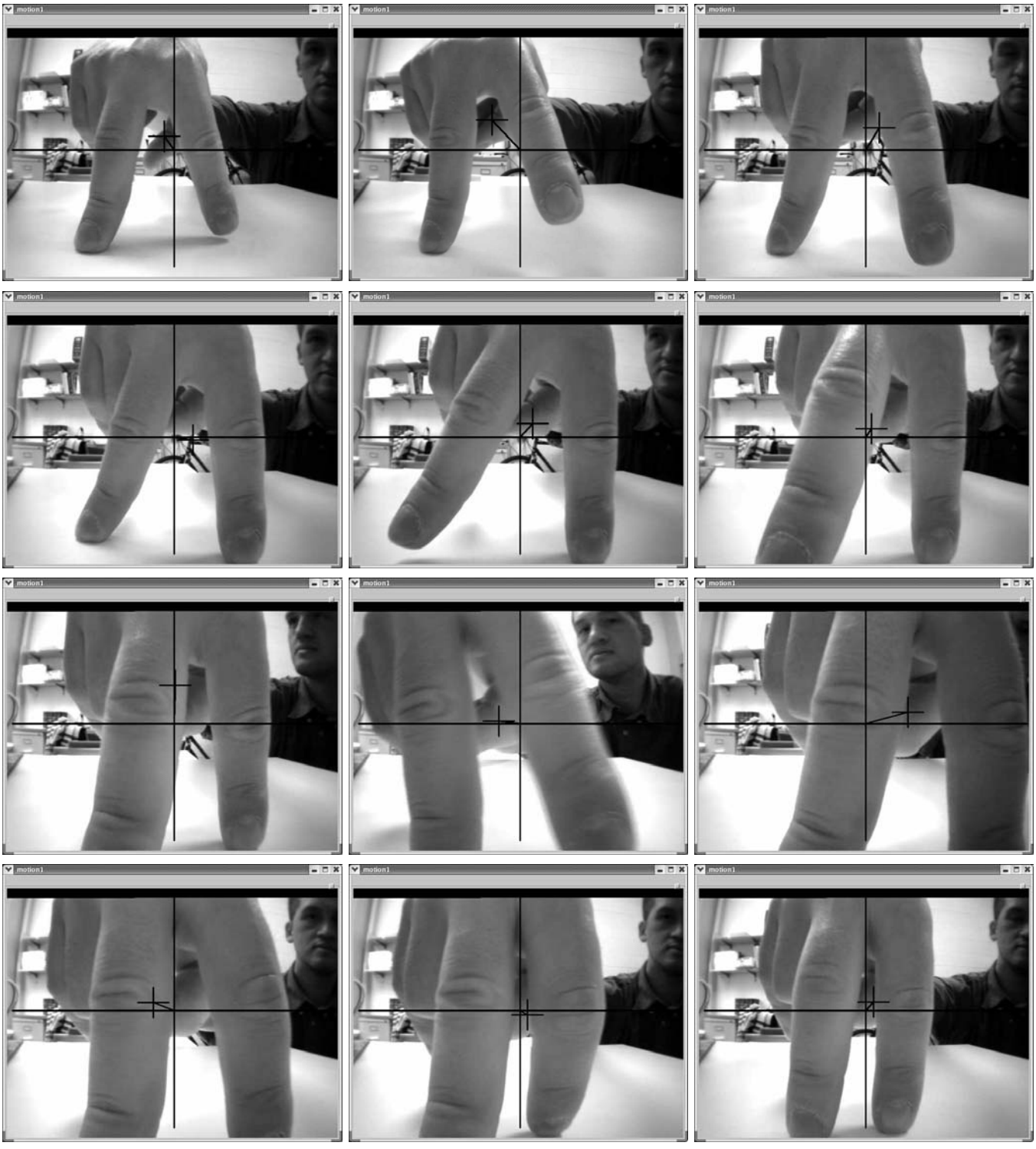


Figure 9: Visual tracking algorithm results

The tracking error obtained from the images is interpreted as shown in Figure 10. It has two components, one of which, Δl_p, is parallel to the pan direction of the camera motion, and the other one, Δl_t, is parallel to the tilt direction of the camera motion. The error values for pan and tilt angles of the camera are inferred from Δl_p and Δl_t respectively, using geometrical arguments.

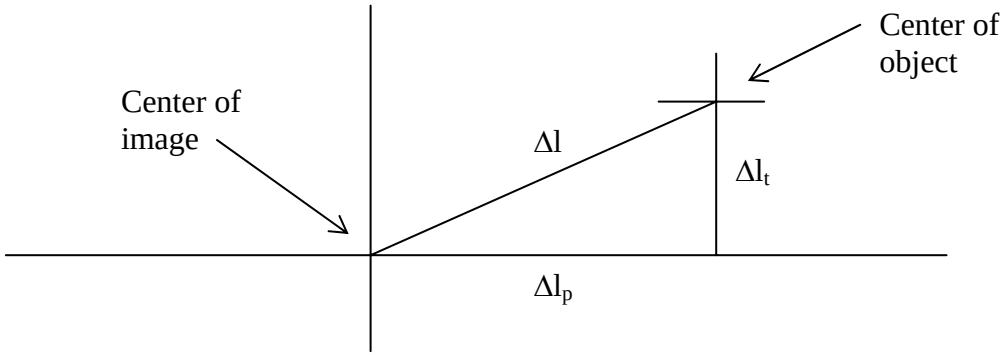


Figure 10: Tracking error components

The implemented visual tracking algorithm performed very well during tests and was able to do tracking an object moving toward camera, which is a more challenging condition for visual tracking than motion in a direction perpendicular to the line of sight. However, the algorithm works only in cases of stationary background and single moving object in the field of view of camera. Thus, its use is limited only to stepwise type of visual tracking described in Figure 5. This makes the algorithm appropriate for using in the simplified problem depicted in Figure 3.

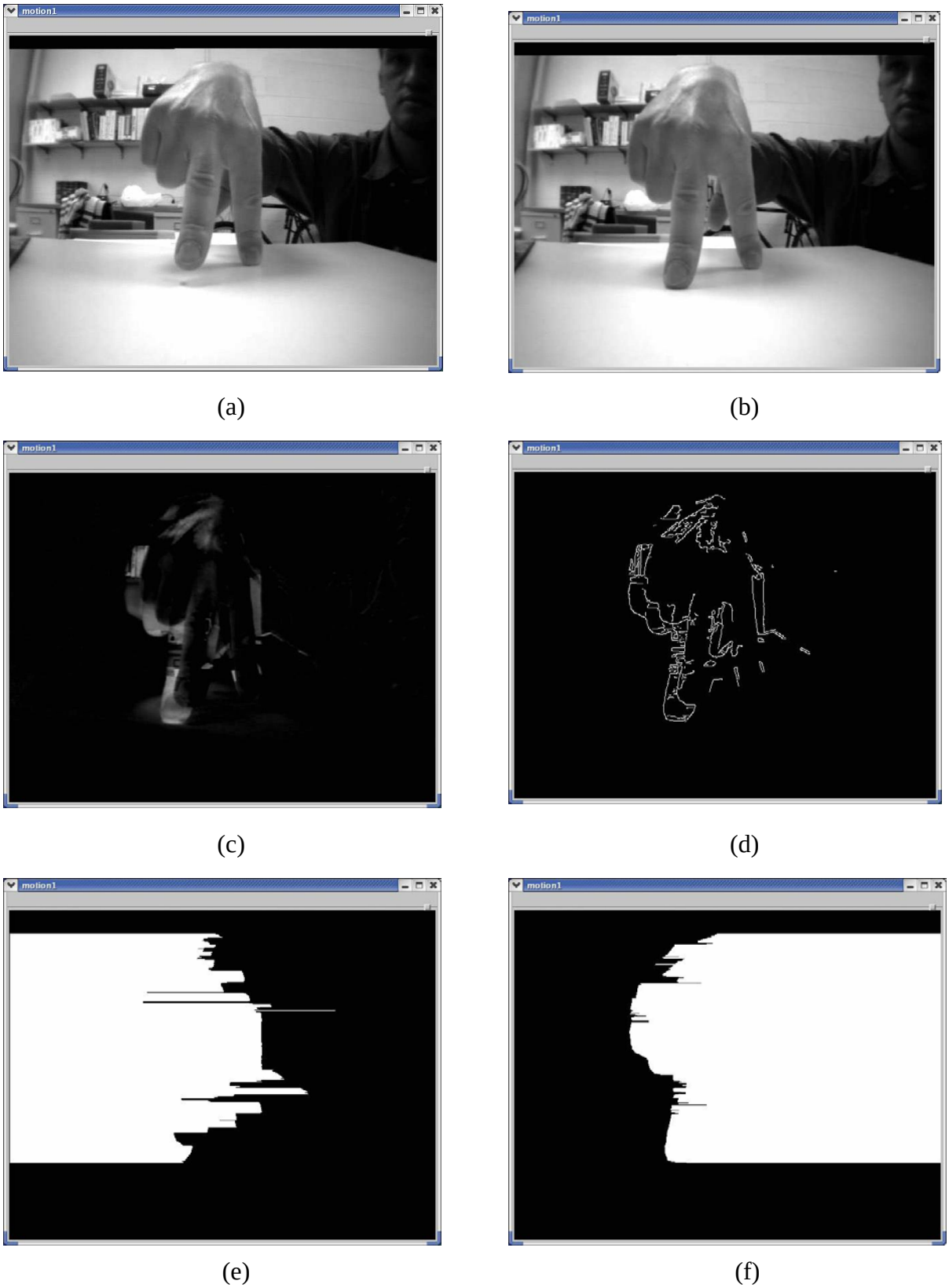
4. Visual Tracking Algorithm Two: Contour Tracking

As in this problem it is necessary to zoom in/out on the object being tracked, tracking algorithms based on object recognition may not be appropriate for use in this problem. Tracking of the center of mass (described in chapter 3) is one class of algorithms that can be used in this problem. Another candidate class of algorithms that can be used in this problem is algorithms based on contour tracking. In a contour tracking algorithm, an object is represented by a contour built around it, and the contour is allowed to change its shape and size. One such algorithm, called Condensation, is described in [2].

In the Condensation algorithm, conditional probability density propagation is used, which is similar to the approach used in the Kalman filter based visual tracking. The problem with the Kalman filter tracking is that the Kalman filter is unimodal, which can lead to its failures in a highly cluttered environment. If a Gaussian of an object from the background is similar to the one that is being tracked by the Kalman filter, the Kalman filter can switch to tracking the object from the background. The Condensation algorithm was designed to overcome this problem by using multimodal tracking. It can track several Gaussians at once, corresponding to several objects. This feature of the Condensation algorithm even allows, according to the authors of the paper, to do visual tracking of an object that becomes occluded for some period of time by another object.

In order to implement contour tracking, first a method should be found to extract a contour from an image that accurately represents a moving object to be tracked. The rest of this section describes an approach for contour extraction that can be used for contour tracking in the simplified problem described in Figure 3.

The contour extraction algorithm works as described in Figure 11 and Figure 12. Images (a) and (b) of Figure 11 are two consecutive images. Images (c) and (d) are the results of the subtraction of images (a) and (b) and edge detection, as before. Image (e) is obtained from image (d) by linewise scanning of image (d) and assigning non-zero values to all pixels after the first found non-zero pixel. Image (f) is created ina similar fashion, moving in the opposite direction.



(a) (b)

(c) (d)

(e) (f)

Figure 11: Contour extraction algorithm

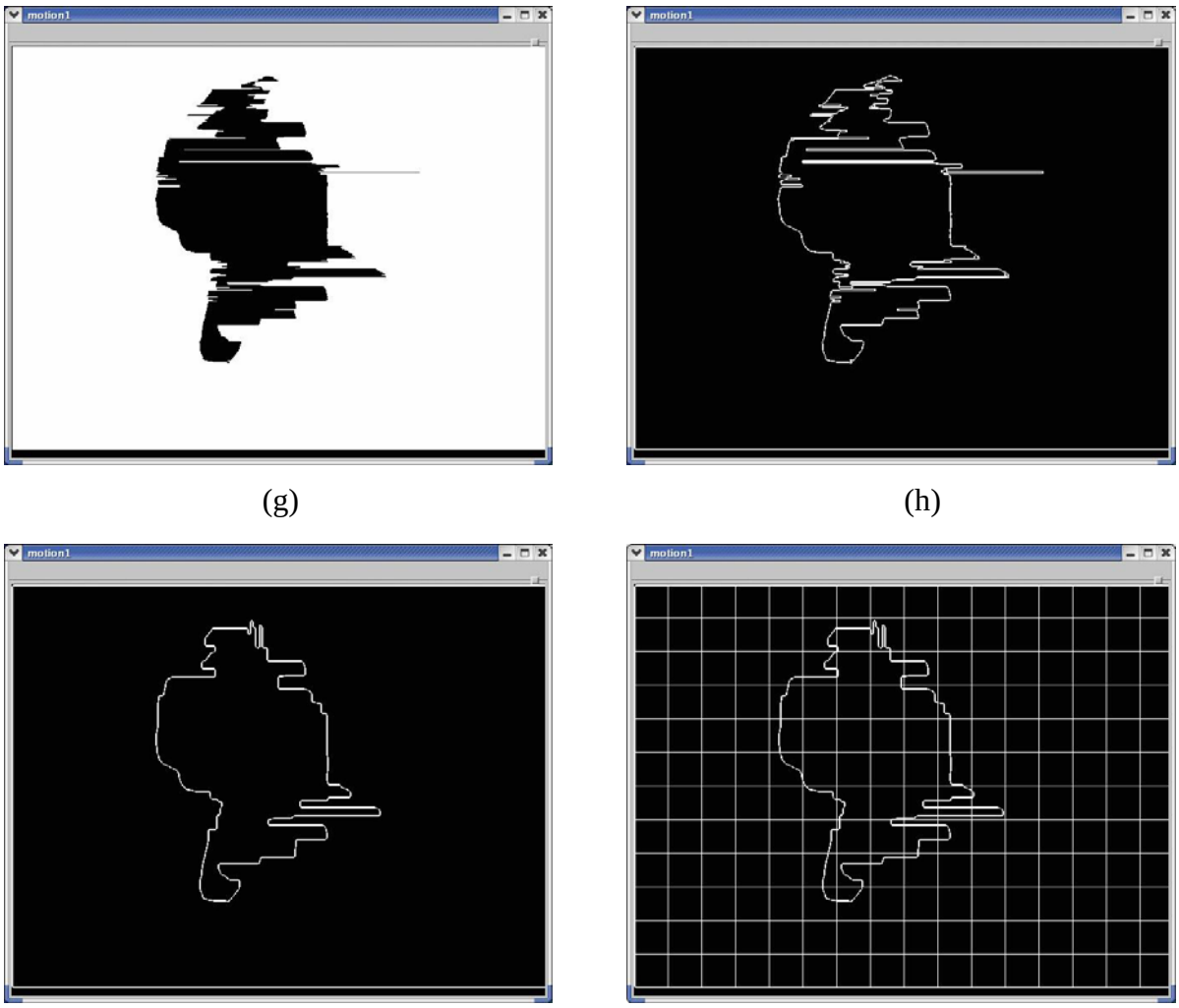


Figure 12: Contour extraction algorithm

Image (g) is obtained by multiplying images (e) and (f). As seen in the figure, the obtained solid object has many spikes. If edge detection algorithm is run on image (g), image (h) is obtained, which does not look like a good contour to track. Therefore algorithm proceeds in a different way. First, a smoothing procedure described below is used to remove spikes in image (g). After that, edge detection algorithm is applied to the resulting smoothed image. The result is image (i). As can be seen in the image, the left side of the obtained contour matches the moving object quite well, unlike the right side of

the contour, that has many spikes on it. This effect takes place for the following reasons. First, there is a window on the left-hand side of the scene, with light incident on the object from left. This results in a better contrast on the left side. Secondly, the hand is not the only moving object in the scene. There is also some motion of the arm, which results in some background motion. In some cases, adjusting threshold values used in the edge detection algorithm allows to suppress the background motion, but not always. If a single object, such as the rover, is moving in a well illuminated scene, this algorithm will produce a contour closely matching the moving object. In order to use the obtained contour in the tracking algorithm, the contour should be sampled, which can be done by superimposing a grid on image (i), as shown in image (j).

The spike smoothing algorithm applied on image (g) before contour extraction is described in Figure 13.

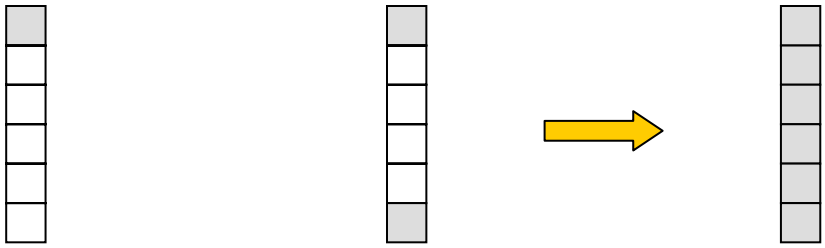


Figure 13: Spike smoothing algorithm

The algorithm works as follows. A window with the width of one pixel and the height of several pixels is used to scan image (g). If the values of the two boundary pixels (the topmost one and the bottommost one) are different, no action is taken. If the values of the boundary pixels are the same, all pixels in between are assigned the same values as the value of the boundary pixels. This procedure allows to get rid of spikes whose thickness is within the vertical size of the window.

As in the case of the tracking algorithm described in the previous section, the use of the contour extraction algorithm is limited to cases of static background and a single

moving object in the field of view of camera. This makes the algorithm appropriate for use in the simplified problem depicted in Figure 3.

At the time of the writing of this paper, this work is in progress, and the next steps will include the implementation of the Condensation algorithm in the experimental setup of the simplified problem using the described contour extraction algorithm.

5. References

[1] B. K. P. Horn, *Robot Vision*, The MIT Press, 1997

[2] M. Isard, A. Blake, *Condensation – conditional density propagation for visual tracking*, Int. J. Computer Vision, 1998